

## Team T

1. Hemanth Thathireddy (A20525346)
2. Satya Dineswara Reddy Satti (A20528639)
3. Sai Manasa Basani (A20544372)
4. Alavalapati Veera manohar Reddy (A20526070)
5. Ying Zhang (A20547843)

1.

a) Find everything found in the Sailors table

b) SQL Command:

**SELECT \* FROM Sailors;**

c) Result:

The screenshot shows the SQL Fiddle interface with the following components:

- SQL Editor:** Contains the SQL command `SELECT * FROM Sailors;`.
- Schema Editor:** Contains the following SQL code:

```
14 (27, 'Adams', 8, 36),
15 (22, 'Carrey', 10, 22);
16
17 -- Create Boats table
18 CREATE TABLE Boats (
19   BID INT PRIMARY KEY,
20   BNAME VARCHAR(50),
21   FEE INT,
22   LOCATION VARCHAR(50)
23 );
24
25 -- Insert data into Boats table
26 INSERT INTO Boats (BID, BNAME, FEE, LOCATION)
27 VALUES
28 (109, 'wayfarer', 120, 'hout bay'),
29 (108, 'seapride', 500, 'fish hook'),
30 (101, 'yuppie', 400, 'hout bay'),
31 (104, 'joy', 200, 'hout bay');
```
- Execution Controls:** Buttons for "Build Schema", "Edit Fullscreen", "Browser", "Run SQL", "Edit Fullscreen", and "Link".
- Result Table:** A table with 4 columns: SID, SNAME, RATING, and AGE. It contains 4 rows of data:

SID	SNAME	RATING	AGE
22	Carrey	10	22
23	Marx	8	52
25	Martin	9	51
27	Adams	8	36
- Footer:** A green bar at the bottom showing "Record Count: 4; Execution Time: 10ms" and links to "View Execution Plan" and "link".

2.

a) Get sailor ID, rank & age of all sailors, ordered from highest to lowest rank.

b) SQL Command:

**SELECT SID, RATING \* 10 AS RANK, AGE FROM Sailors ORDER BY RANK DESC;**

c) Result:

The screenshot shows a SQL IDE with two panes. The left pane contains SQL code for creating a 'Boats' table and inserting data. The right pane shows a query to select sailors ranked by rating. Below the panes is a table with 3 columns: SID, RANK, and AGE.

```
14 (27, 'Adams', 8, 36),
15 (22, 'Carrey', 10, 22);
16
17 -- Create Boats table
18 CREATE TABLE Boats (
19   BID INT PRIMARY KEY,
20   BNAME VARCHAR(50),
21   FEE INT,
22   LOCATION VARCHAR(50)
23 );
24
25 -- Insert data into Boats table
26 INSERT INTO Boats (BID, BNAME, FEE, LOCATION)
27 VALUES
28 (109, 'wayfarer', 120, 'hout bay'),
29 (108, 'seapride', 500, 'fish hoek'),
30 (101, 'yuppie', 400, 'hout bay'),
31 (104, 'joy', 200, 'hout bay');
```

```
1 SELECT SID, RATING * 10 AS RANK, AGE FROM Sailors ORDER BY RANK DESC;
```

SID	RANK	AGE
22	100	22
25	90	51
23	80	52
27	80	36

Record Count: 4; Execution Time: 15ms

**Algebra expression:**  $\pi(\text{SID}, \text{RATING} * 10, \text{AGE})\sigma(\text{RATING})(\text{SAILORS})$

3.

a) Get alphabetical list of sailors with rating less than 10.

b) SQL Command:

**SELECT SNAME FROM Sailors WHERE RATING < 10 ORDER BY SNAME;**

c) Result:

The screenshot shows a SQL IDE with two panes. The left pane contains SQL code for creating a 'Boats' table and inserting data. The right pane shows a query to select sailors with a rating less than 10. Below the panes is a table with 1 column: SNAME.

```
14 (27, 'Adams', 8, 36),
15 (22, 'Carrey', 10, 22);
16
17 -- Create Boats table
18 CREATE TABLE Boats (
19   BID INT PRIMARY KEY,
20   BNAME VARCHAR(50),
21   FEE INT,
22   LOCATION VARCHAR(50)
23 );
24
25 -- Insert data into Boats table
26 INSERT INTO Boats (BID, BNAME, FEE, LOCATION)
27 VALUES
28 (109, 'wayfarer', 120, 'hout bay'),
29 (108, 'seapride', 500, 'fish hoek'),
30 (101, 'yuppie', 400, 'hout bay'),
31 (104, 'joy', 200, 'hout bay');
```

```
1 SELECT SNAME FROM Sailors WHERE RATING < 10 ORDER BY SNAME;
```

SNAME
Adams
Martin
Marx

Record Count: 3; Execution Time: 8ms

4.

a) Find how much deposit money there is in total and how many tuples are in the Reserves table.

b) SQL Command:

**SELECT SUM(DEPOSIT) AS TOTAL\_DEPOSIT, COUNT(\*) AS NUM\_TUPLES FROM Reserves;**

c) Result:

The screenshot shows a SQL Fiddle interface with a MySQL 5.6 database. The left pane contains SQL code for creating a 'Boats' table and inserting data. The right pane contains a query: `SELECT SUM(DEPOSIT) AS TOTAL_DEPOSIT, COUNT(*) AS NUM_TUPLES FROM Reserves;`. Below the code panes, a table displays the query results:

TOTAL_DEPOSIT	NUM_TUPLES
220	4

At the bottom, a status bar indicates: Record Count: 1; Execution Time: 17ms. Links for 'View Execution Plan' and 'link' are also present.

5.

a) Get all info on boats in locations similar to "Fishhoek."

b) SQL Command:

**SELECT \* FROM Boats WHERE LOCATION LIKE '%IS%K';**

c) Result:

The screenshot shows a SQL Fiddle interface with a MySQL 5.6 database. The left pane contains the same SQL code as the previous screenshot. The right pane contains a query: `SELECT * FROM Boats WHERE LOCATION LIKE '%IS%K';`. Below the code panes, a table displays the query results:

BID	BNAME	FEE	LOCATION
108	seapride	500	fish hoek

At the bottom, a status bar indicates: Record Count: 1; Execution Time: 1ms. Links for 'View Execution Plan' and 'link' are also present.

6.

a) Get distinct locations where boats are kept.

b) SQL Command:

**SELECT DISTINCT LOCATION FROM Boats;**

c) Result:

The screenshot shows a SQL IDE with a script editor on the left and a query editor on the right. The script editor contains the following SQL code:

```
14 (27, 'Adams', 8, 36),
15 (22, 'Carrey', 10, 22);
16
17 -- Create Boats table
18 CREATE TABLE Boats (
19   BID INT PRIMARY KEY,
20   BNAME VARCHAR(50),
21   FEE INT,
22   LOCATION VARCHAR(50)
23 );
24
25 -- Insert data into Boats table
26 INSERT INTO Boats (BID, BNAME, FEE, LOCATION)
27 VALUES
28 (109, 'wayfarer', 120, 'hout bay'),
29 (108, 'seapride', 500, 'fish hook'),
30 (101, 'yuppie', 400, 'hout bay'),
31 (104, 'joy', 200, 'hout bay');
```

The query editor on the right contains the query: `SELECT DISTINCT LOCATION FROM Boats;`

Below the editors, there are buttons: "Build Schema", "Edit Fullscreen", "Browser", and "Run SQL". The "Run SQL" button is highlighted.

The result of the query is displayed in a table below the buttons:

LOCATION
hout bay
fish hook

At the bottom, a green status bar shows: "Record Count: 2; Execution Time: 8ms" with links to "View Execution Plan" and "link".

7.

a) Get the names of all boats that have a fee value recorded in the database.

b) SQL Command:

**SELECT BNAME FROM Boats WHERE FEE IS NOT NULL;**

c) Result:

The screenshot shows a SQL IDE with a script editor on the left and a query editor on the right. The script editor contains the same SQL code as in the previous screenshot.

The query editor on the right contains the query: `SELECT BNAME FROM Boats WHERE FEE IS NOT NULL;`

Below the editors, there are buttons: "Build Schema", "Edit Fullscreen", "Browser", and "Run SQL". The "Run SQL" button is highlighted.

The result of the query is displayed in a table below the buttons:

BNAME
yuppie
joy
seapride
wayfarer

8.

a) Get ID of all boats that have not been reserved.

b) SQL Command:

**SELECT BID FROM Boats WHERE BID NOT IN (SELECT BID FROM Reserves);**

c) Result:

The screenshot shows a SQL Fiddle interface with the following SQL code:

```

14 (27, 'Adams', 8, 36),
15 (22, 'Carrey', 10, 22);
16
17 -- Create Boats table
18 CREATE TABLE Boats (
19   BID INT PRIMARY KEY,
20   BNAME VARCHAR(50),
21   FEE INT,
22   LOCATION VARCHAR(50)
23 );
24
25 -- Insert data into Boats table
26 INSERT INTO Boats (BID, BNAME, FEE, LOCATION)
27 VALUES
28 (109, 'wayfarer', 120, 'hout bay'),
29 (108, 'seapride', 500, 'fish hoek'),
30 (101, 'yuppie', 400, 'hout bay'),
31 (104, 'joy', 200, 'hout bay');

```

The query in the right pane is:

```

1 SELECT BID FROM Boats WHERE BID NOT IN (SELECT BID FROM Reserves);

```

The execution plan shows the following details:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	Boats	index		PRIMARY	4		4	100.00	Using where; Using index
2	DEPENDENT SUBQUERY	Reserves	index_subquery	BID	BID	4	func	1	100.00	Using index

**Algebra expression:**  $\pi(\text{BOAT\_ID})(\text{BOATS} - (\sigma(\text{BOAT\_ID} = \text{RESERVES.BOAT\_ID})(\text{RESERVES})))$

9.

a) Get all reservation info, including all details on the boats being reserved.

b) SQL Command:

**SELECT \* FROM Reserves INNER JOIN Boats ON Reserves.BID = Boats.BID;**

c) Result:

The screenshot shows a SQL Fiddle interface with the following SQL code:

```

14 (27, 'Adams', 8, 36),
15 (22, 'Carrey', 10, 22);
16
17 -- Create Boats table
18 CREATE TABLE Boats (
19   BID INT PRIMARY KEY,
20   BNAME VARCHAR(50),
21   FEE INT,
22   LOCATION VARCHAR(50)
23 );
24
25 -- Insert data into Boats table
26 INSERT INTO Boats (BID, BNAME, FEE, LOCATION)
27 VALUES
28 (109, 'wayfarer', 120, 'hout bay'),
29 (108, 'seapride', 500, 'fish hoek'),
30 (101, 'yuppie', 400, 'hout bay'),
31 (104, 'joy', 200, 'hout bay');

```

The query in the right pane is:

```

1 SELECT * FROM Reserves INNER JOIN Boats ON Reserves.BID = Boats.BID;

```

The result set shows the following data:

SID	BID	DAY	DEPOSIT	BID	BNAME	FEE	LOCATION
25	101	2023-08-29	50	101	yuppie	400	hout bay
23	104	2023-08-29	0	104	joy	200	hout bay
27	108	2023-08-29	70	108	seapride	500	fish hoek
22	109	2023-08-29	100	109	wayfarer	120	hout bay

**Algebra expression:**  $\rho(\text{ReservationInfo}, (\sigma(\text{RESERVES.BOAT\_ID} = \text{BOATS.BOAT\_ID})(\text{RESERVES} \bowtie \text{BOATS})))$

10.

a) For all reservations, get the name of the sailor, along with the day and name of the boat booked.

b) SQL Command:

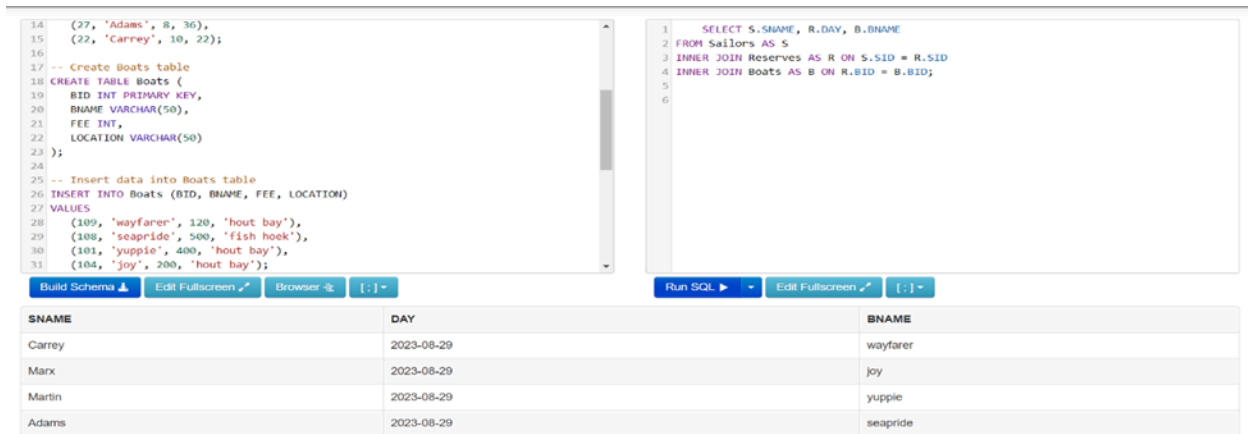
```
SELECT S.SNAME, R.DAY, B.BNAME
```

```
FROM Sailors AS S
```

```
INNER JOIN Reserves AS R ON S.SID = R.SID
```

```
INNER JOIN Boats AS B ON R.BID = B.BID;
```

c) Result:



```
14 (27, 'Adams', 8, 36),
15 (22, 'Carrey', 10, 22);
16
17 -- Create Boats table
18 CREATE TABLE Boats (
19   BID INT PRIMARY KEY,
20   BNAME VARCHAR(50),
21   FEE INT,
22   LOCATION VARCHAR(50)
23 );
24
25 -- Insert data into Boats table
26 INSERT INTO Boats (BID, BNAME, FEE, LOCATION)
27 VALUES
28   (109, 'wayfarer', 120, 'hout bay'),
29   (108, 'seapride', 500, 'fish hook'),
30   (101, 'yuppie', 400, 'hout bay'),
31   (104, 'joy', 200, 'hout bay');
```

```
1 SELECT S.SNAME, R.DAY, B.BNAME
2 FROM Sailors AS S
3 INNER JOIN Reserves AS R ON S.SID = R.SID
4 INNER JOIN Boats AS B ON R.BID = B.BID;
5
6
```

SNAME	DAY	BNAME
Carrey	2023-08-29	wayfarer
Marx	2023-08-29	joy
Martin	2023-08-29	yuppie
Adams	2023-08-29	seapride

**Algebra expression:**  $\rho(\text{ReservationDetails},$   
 $(\sigma(\text{SAILORS.SID} = \text{RESERVES.SID}) \bowtie (\text{RESERVES} \bowtie \text{BOATS})))$

11. Get the average deposit paid for each boat.

```
SELECT BID, AVG(deposit) FROM Reserves GROUP BY BID
```

The screenshot shows a database management tool interface. On the left, a sidebar lists the database structure under 'SailingDB', including Tables (Boats, Captains, Reserves, Sailors), Views, Stored Procedures, Functions, and a 'sys' system database. The main area displays a query: `SELECT BID, AVG(deposit) FROM Reserves GROUP BY BID`. Below the query, a 'Result Grid' shows the output of the query. The grid has two columns: 'BID' and 'AVG(deposit)'. The results are as follows:

BID	AVG(deposit)
101	50.000000
104	0.000000
108	120.000000
109	80.000000

Below the result grid, an 'Action Output' section shows the execution log:

	Time	Action	Response	Duration / Fetch Time
146	22:10:39	DROP DATABASE 'HEM'	4 row(s) affected	0.068 sec
147	22:13:23	SELECT BID, AVG(deposit) FROM Reserves GROUP BY BID	4 row(s) returned	0.0059 sec / 0.00001...

The status bar at the bottom indicates 'Query Completed'.

12. Get the average deposit paid for each boat that has been booked by more than one person.

**SELECT BID, AVG(deposit) FROM Reserves GROUP BY BID HAVING  
COUNT(DISTINCT SID) > 1**

The screenshot shows a database management tool interface. The left sidebar displays a schema tree for 'SailingDB' with tables: Boats, Captains, Reserves, Sailors, Views, Stored Procedures, Functions, and sys. The main query editor shows a query: `SELECT BID, AVG(deposit) FROM Reserves GROUP BY BID HAVING COUNT(DISTINCT SID) > 1`. The 'Result Grid' shows two rows of data:

BID	AVG(deposit)
101	50.000000
109	80.000000

The 'Action Output' pane shows the execution log:

	Time	Action	Response	Duration / Fetch Time
12	22:45:00	SELECT BID, AVG(deposit) FROM Rese...	4 row(s) returned	0.00060 sec / 0.000...
13	22:46:52	SELECT BID, AVG(deposit) FROM Rese...	2 row(s) returned	0.0036 sec / 0.00001...

Query Completed

13. Get the average firm deposit paid for each boat that has been booked by more than one person, in increasing order of amount. A firm deposit is one which exceeds R10.

**SELECT BID, AVG(deposit) AS AVERAGEdeposit FROM Reserves WHERE DEPOSIT > 10 GROUP BY BID HAVING COUNT(DISTINCT SID) > 1 ORDER BY AVERAGEdeposit**

The screenshot shows the same database management tool interface. The query editor now contains: `SELECT BID, AVG(deposit) AS AVERAGEdeposit FROM Reserves WHERE DEPOSIT > 10 GROUP BY BID HAVING COUNT(DISTINCT SID) > 1 ORDER BY AVERAGEdeposit`. The 'Result Grid' shows one row of data:

BID	AVERAGEdeposit
109	120.000000

The 'Action Output' pane shows the execution log:

	Time	Action	Response	Duration / Fetch Time
14	22:49:32	SELECT BID, AVG(deposit) AS AVERAG...	1 row(s) returned	0.0029 sec / 0.00001...
15	22:49:52	SELECT BID, AVG(deposit) AS AVERAG...	1 row(s) returned	0.00060 sec / 0.000...

Query Completed



14. Get name & rating of sailors with rating exceeding 7 who made any reservation with 0 deposit.

**SELECT Sname, rating FROM Sailors WHERE rating > 7 AND SID IN  
(SELECT SID FROM Reserves WHERE deposit = 0)**

Query execution results for problem 14:

Sname	rating
Martin	9
Carrey	10

Action Output:

Time	Action	Response	Duration / Fetch Time
15 22:49:52	SELECT BID, AVG(deposit) AS AVERAG...	1 row(s) returned	0.00060 sec / 0.000...
16 19:15:13	SELECT Sname, rating FROM Sailors W...	2 row(s) returned	0.106 sec / 0.00053...

15. Get names of boats located in a place other than Hout Bay or Fish Hoek.

**SELECT Bname FROM Boats WHERE location NOT IN ("Hout Bay", "Fish Hoek")**

Query execution results for problem 15:

Bname
-------

Action Output:

Time	Action	Response	Duration / Fetch Time
17 19:18:30	SELECT * FROM SailingDB.Boats LIMIT...	4 row(s) returned	0.0072 sec / 0.00001...
18 19:19:07	SELECT Bname FROM Boats WHERE lo...	0 row(s) returned	0.0031 sec / 0.00000...

16. Get names of boats having a fee larger than any boat located in Hout Bay.

**SELECT DISTINCT Bname FROM Boats WHERE fee > SOME  
(SELECT fee FROM Boats WHERE location = "Hout Bay")**

The screenshot shows a database management tool interface. The left sidebar displays the 'Schemas' tree with 'SailingDB' expanded, showing tables like 'Boats', 'Captains', 'Reserves', and 'Sailors'. The main query editor contains two lines of SQL:   
1. `SELECT DISTINCT Bname FROM Boats WHERE fee > SOME`  
2. `(SELECT fee FROM Boats WHERE location = "Hout Bay")`  
The 'Result Grid' shows the results of the query, with columns 'Bname' and 'fee'. The results are:   
| Bname | fee |  
|---|---|  
| Yuppie | 10.0 |  
| Joy | 10.0 |  
| SeaPride | 10.0 |  
The 'Action Output' table at the bottom shows the execution details:   
| Time | Action | Response | Duration / Fetch Time | |
|---|---|---|---|---|
| 18 | 19:19:07 | SELECT Bname FROM Boats WHERE lo... | 0 row(s) returned | 0.0031 sec / 0.00000... |  
| 19 | 19:21:56 | SELECT DISTINCT Bname FROM Boats... | 3 row(s) returned | 0.012 sec / 0.000008... |

17. Get names that are in both the sailors and the captains relations

**SELECT Sname FROM Sailors WHERE EXISTS  
(SELECT \* FROM Captains WHERE Captains.SID = Sailors. SID)**

The screenshot shows the same database management tool interface. The left sidebar shows the 'Schemas' tree with 'SailingDB' expanded. The main query editor contains two lines of SQL:   
1. `SELECT Sname FROM Sailors WHERE EXISTS`  
2. `(SELECT * FROM Captains WHERE Captains.SID = Sailors. SID)`  
The 'Result Grid' shows the results of the query, with columns 'Sname' and 'SID'. The results are:   
| Sname | SID |  
|---|---|  
| Marx | 100 |  
| Martin | 101 |  
| Adams | 102 |  
| Carrey | 103 |  
The 'Action Output' table at the bottom shows the execution details:   
| Time | Action | Response | Duration / Fetch Time | |
|---|---|---|---|---|
| 19 | 19:21:56 | SELECT DISTINCT Bname FROM Boats... | 3 row(s) returned | 0.012 sec / 0.000008... |  
| 20 | 19:24:07 | SELECT Sname FROM Sailors WHERE E... | 4 row(s) returned | 0.0014 sec / 0.00000... |

**Algebra expression:**  $\pi \text{ Sname (Sailors)} \cap \pi \text{ Sname (Captains)}$

18. Get names of boats that have exactly 1 reservation.

**SELECT B.BNAME**

**FROM Boats AS B**

**WHERE B.BID IN (**

**SELECT R.BID**

**FROM Reserves AS R**

**GROUP BY R.BID**

**HAVING COUNT(\*) = 1**

**);**

The screenshot shows a SQL IDE interface. On the left, a SQL script is visible, including table creation and data insertion for a 'Boats' table. On the right, the same query as in the previous block is entered. Below the query editor, there are buttons for 'Build Schema', 'Edit Fullscreen', 'Browser', and 'Run SQL'. The 'Run SQL' button is highlighted, and the results of the query are displayed in a table below. The table has a single column 'BNAME' and four rows of data: 'yuppie', 'joy', 'seapride', and 'wayfarer'. At the bottom, a status bar indicates 'Record Count: 4; Execution Time: 4ms' and provides links to 'View Execution Plan' and 'link'.

```
14 (27, 'Adams', 8, 36),
15 (22, 'Carrey', 10, 22);
16
17 -- Create Boats table
18 CREATE TABLE Boats (
19   BID INT PRIMARY KEY,
20   BNAME VARCHAR(50),
21   FEE INT,
22   LOCATION VARCHAR(50)
23 );
24
25 -- Insert data into Boats table
26 INSERT INTO Boats (BID, BNAME, FEE, LOCATION)
27 VALUES
28 (109, 'wayfarer', 120, 'hout bay'),
29 (108, 'seapride', 500, 'fish hoek'),
30 (101, 'yuppie', 400, 'hout bay'),
31 (104, 'joy', 200, 'hout bay');
```

```
1 SELECT B.BNAME
2 FROM Boats AS B
3 WHERE B.BID IN (
4   SELECT R.BID
5   FROM Reserves AS R
6   GROUP BY R.BID
7   HAVING COUNT(*) = 1
8 );
9
10
```

BNAME
yuppie
joy
seapride
wayfarer

✓ Record Count: 4; Execution Time: 4ms   [View Execution Plan](#)   [link](#)

19. Get sailor ID and total deposit paid for sailors who have booked more than 1 boat

**SELECT SID, TOTAL\_deposit FROM (SELECT SID, COUNT(BID),  
SUM(deposit) FROM Reserves WHERE deposit IS NOT NULL AND deposit > 0  
GROUP BY SID ) AS RESULT(SID, NUM\_Boats, TOTAL\_deposit) WHERE NUM\_Boats > 1**

Administration Schemas HomeWork1.1\* Reserves Boats

SCHEMAS

Filter objects

SailingDB

- Tables
  - Boats
  - Captains
  - Reserves
  - Sailors
- Views
- Stored Procedures
- Functions
- sys

Object Info Session

Table: Boats

Columns:

- BID int PK
- Bname varchar(100)
- fee decimal(10,2)
- location varchar(100)

```

1 SELECT SID, TOTAL_deposit FROM (SELECT SID, COUNT(BID),
2 SUM(deposit) FROM Reserves WHERE deposit IS NOT NULL AND deposit > 0
3 GROUP BY SID ) AS RESULT(SID, NUM_Boats, TOTAL_deposit) WHERE NUM_Boats > 1

```

100% 1:1

Result Grid

SID	TOTAL_deposit
23	240.00
27	220.00

Result 10

Action Output

	Time	Action	Response	Duration / Fetch Time
25	19:35:58	SELECT SID, TOTAL_deposit FROM (SE...	Error Code: 1146, Table 'sailingdb.reserves' doesn't ex...	0.0010 sec
26	19:36:07	SELECT SID, TOTAL_deposit FROM (SE...	2 row(s) returned	0.0051 sec / 0.00001...

Query Completed

20. Get all reservation info including details of the boat booked.

**SELECT \* FROM Boats INNER JOIN Reserves ON Boats.BID = Reserves.BID**

Administration Schemas HomeWork1.1\* Reserves Boats

SCHEMAS

Filter objects

SailingDB

- Tables
  - Boats
  - Captains
  - Reserves
  - Sailors
- Views
- Stored Procedures
- Functions
- sys

Object Info Session

Table: Boats

Columns:

- BID int PK
- Bname varchar(100)
- fee decimal(10,2)
- location varchar(100)

```

1 SELECT * FROM Boats INNER JOIN Reserves ON Boats.BID = Reserves.BID

```

100% 68:1

Result Grid

BID	Bname	fee	location	SID	BID	day	deposit
101	Yuppie	400.00	Hout Bay	25	101	2014-08-08	0.00
101	Yuppie	400.00	Hout Bay	27	101	2014-08-09	100.00
104	Joy	200.00	Hout Bay	33	104	2014-09-11	0.00
108	SeaPride	500.00	Fish Hock	23	108	2014-08-08	120.00
109	Wayfarer	120.00	Hout Bay	23	109	2014-08-01	120.00
109	Wayfarer	120.00	Hout Bay	27	109	2014-08-15	120.00
109	Wayfarer	120.00	Hout Bay	33	109	2014-09-04	0.00

Result 11

Action Output

	Time	Action	Response	Duration / Fetch Time
26	19:36:07	SELECT SID, TOTAL_deposit FROM (SE...	2 row(s) returned	0.0051 sec / 0.00001...
27	19:39:32	SELECT * FROM Boats INNER JOIN Res...	7 row(s) returned	0.0035 sec / 0.00001...

Query Completed

**Algebra expression:**  $\rho(\text{ReservationInfo}, (\sigma(\text{Reserves.Boat\_ID} = \text{Boats.Boat\_ID})(\text{Reserves} \bowtie \text{Boats})))$

21.

a) Get all information on every boat. If a boat has reservations, include all its reservation info.

b) SQL Command:

**SELECT \* FROM Boats LEFT OUTER JOIN Reserves on Boats.Bid= Reserves.Bid**

c) Result:

21 Input

SELECT \* FROM Boats LEFT OUTER JOIN Reserves on Boats.BID = Reserves.BID

Run SQL

Output

BID	Bname	fee	location	SID	BID	day	deposit
109	Wayfarer	120	Hout Bay	23	109	2014-08-01	120
109	Wayfarer	120	Hout Bay	27	109	2014-08-15	120
109	Wayfarer	120	Hout Bay	33	109	2014-09-04	0
108	SeaPride	500	Fish Hoek	23	108	2014-08-08	120
101	Yuppie	400	Hout Bay	25	101	2014-08-08	0
101	Yuppie	400	Hout Bay	27	101	2014-08-09	100
104	Joy	200	Hout Bay	33	104	2014-09-11	0

22.

a) Create a new tuple for the boat named “Nino” which has fee R150, BID 110, and is in Fish Hoek.

b) SQL Command:

**INSERT INTO Boats (BNAME, BID, FEE, LOCATION)**

**VALUES ('Nino', 110, 150, 'Fish Hoek');**

c) Result:

22 Input

INSERT INTO Boats VALUES ('Nino',110,150,'Fish Hoek')

Run SQL

Available Tables

Boats

BID	Bname	fee	location
109	Wayfarer	120	Hout Bay
108	SeaPride	500	Fish Hoek
101	Yuppie	400	Hout Bay
104	Joy	200	Hout Bay
Nino	110	150	Fish Hoek

23.

a) Remove all bookings from Reserves where there is no deposit.

b) SQL Command:

**DELETE FROM Reserves**

**WHERE DEPOSIT IS NULL OR DEPOSIT = 0;**

c) Result:

The screenshot shows a SQL IDE interface. On the left, the 'Input' tab contains the SQL command: `DELETE FROM Reserves WHERE deposit is NULL OR deposit = 0`. A 'Run SQL' button is visible. On the right, the 'Available Tables' section shows the 'Reserves' table. Below it, a preview of the 'Reserves' table is displayed with the following data:

SID	BID	day	deposit
23	109	2014-08-01	120
23	108	2014-08-08	120
27	101	2014-08-09	100
27	109	2014-08-15	120

24.

a) Increase the fee of every boat by 12%.

b) SQL Command:

**UPDATE Boats**

**SET FEE = FEE \* 1.12;**

c) Result:

The screenshot shows a SQL IDE interface. On the left, the 'Input' tab contains the SQL command: `UPDATE Boats SET fee = fee*1.12`. A 'Run SQL' button is visible. On the right, the 'Available Tables' section shows the 'Boats' table. Below it, a preview of the 'Boats' table is displayed with the following data:

BID	Bname	fee	location
109	Wayfarer	134.4	Hout Bay
108	SeaPride	560	Fish Hoek
101	Yuppie	448.000000000000006	Hout Bay
104	Joy	224.000000000000003	Hout Bay
Nino	110	168.000000000000003	Fish Hoek

25.

a) Make a view called Bookings which hides the Deposit value i.e. only has the other 3 attributes.

b) SQL Command:

**CREATE VIEW Bookings AS**

**SELECT SID, BID, DAY**

**FROM Reserves;**

c) Result:

25 Input ↺ ↻ ⋮ Run SQL > Available Tables

```
CREATE VIEW BOOKINGS AS SELECT SID,BID,day FROM Reserves
```

Reserves

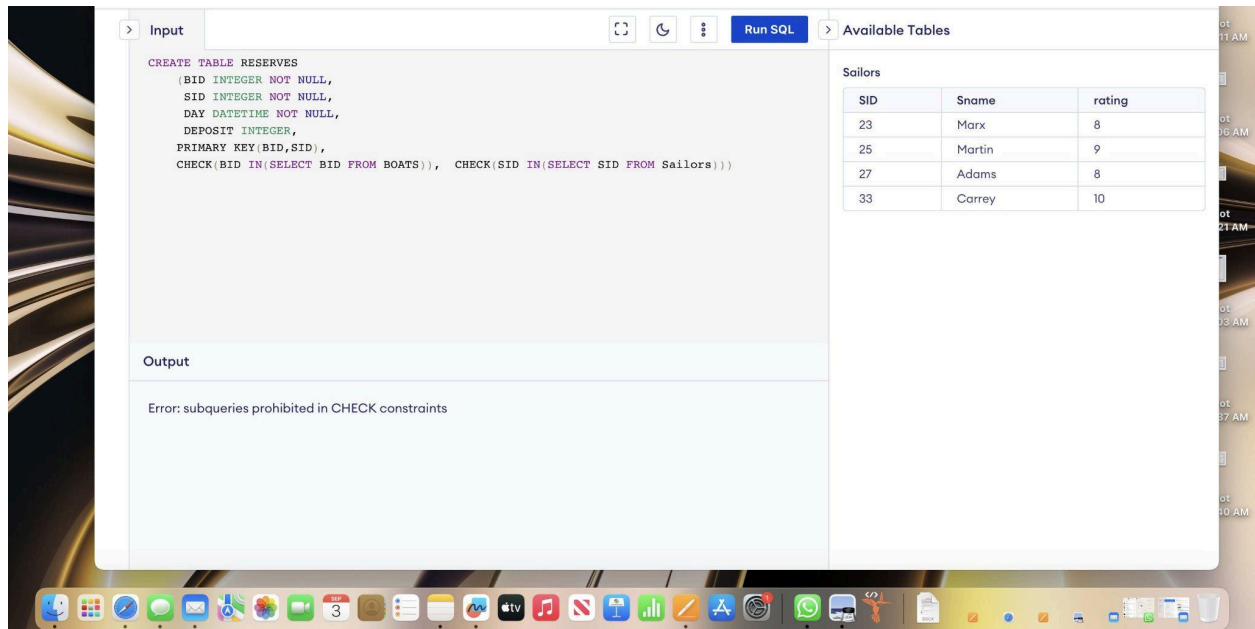
SID	BID	day	deposit
23	109	2014-08-01	120
23	108	2014-08-08	120
27	101	2014-08-09	100
27	109	2014-08-15	120

26. Create a table called Reserves with 3 integer attributes BID, SID & deposit, and a date attribute Day. Allow only deposit to be omitted, and ensure SID and BID values exist in the database. When someone books a boat it is for the whole day.

b) SQL Command:

```
CREATE TABLE Reserves (  
  
    BID INTEGER NOT NULL,  
  
    SID INTEGER NOT NULL,  
  
    DAY DATETIME NOT NULL,  
  
    DEPOSIT INTEGER,  
  
    PRIMARY KEY (BID, SID),  
  
    CHECK (BID IN (SELECT BID FROM Boats)),  
  
    CHECK (SID IN (SELECT SID FROM Sailors))  
  
);
```

c) Result:



27.

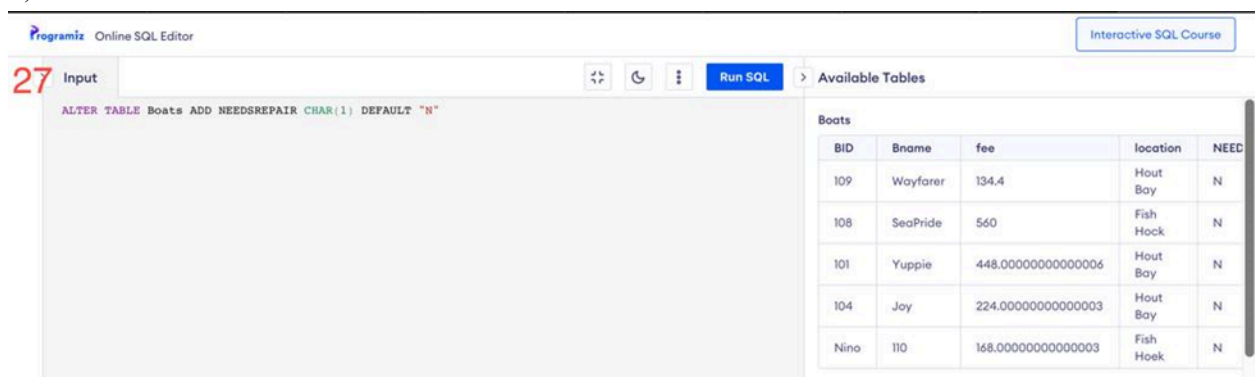
a) Add a new attribute NEEDSREPAIR to the Boats table. It is usually "N".

b) SQL Command:

**ALTER TABLE Boats**

**ADD NEEDSREPAIR CHAR(1) DEFAULT 'N';**

c) Result:



28. We should not be ageist. Remove the Age attribute.



a) Remove the Age attribute from the Sailors table.

b) SQL Command:

**ALTER TABLE Sailors DROP AGE;**

c) Result:

28

Input

```
ALTER TABLE Sailors DROP AGE
```

Run SQL

Available Tables

Sailors

SID	Sname	rating
23	Marx	8
25	Martin	9
27	Adams	8
33	Correy	10

29.

a) Remove the Captains relation altogether so that nobody can try insert or use Captains in future.

b) SQL Command:

**DROP TABLE Captains;**

c) Result:

> Input

```
DROP TABLE Captains
```

Run SQL

Output

SQL query successfully executed. However, the result set is empty.

## Team Contributions:

S.No	Name	Contribution(%)
1.	Hemanth Thathireddy	20%
2.	Satya Dineswara Reddy Setti	20%
3.	Sai Manasa Basani	20%
4.	Alavalapati Veera manohar Reddy	20%
5.	Ying Zhang	20%